



US 20020146123A1

**(19) United States**

**(12) Patent Application Publication**  
Tian

**(10) Pub. No.: US 2002/0146123 A1**  
**(43) Pub. Date: Oct. 10, 2002**

**(54) CONTENT AUTHENTICATION AND  
RECOVERY USING DIGITAL WATERMARKS**

on Jan. 10, 2001. Provisional application No. 60/284,  
594, filed on Apr. 17, 2001.

**(76) Inventor: Jun Tian, Tualatin, OR (US)**

**Publication Classification**

Correspondence Address:  
**DIGIMARC CORPORATION**  
19801 SW 72ND AVENUE  
SUITE 100  
TUALATIN, OR 97062 (US)

**(51) Int. Cl.<sup>7</sup> .....** H04N 7/167  
**(52) U.S. Cl. .....** 380/234

**(21) Appl. No.: 10/035,830**

**(57) ABSTRACT**

**(22) Filed: Oct. 18, 2001**

The disclosure describes methods for using digital watermarking to authenticate digital media signals, such as images, audio and video signals. It also describes techniques for using embedded watermarks to repair altered parts of a media signal when alteration is detected. Alteration is detected using hashes, digital watermarks, and a combination of hashes and digital watermarks.

**Related U.S. Application Data**

**(60) Provisional application No. 60/247,389, filed on Nov.  
8, 2000. Provisional application No. 60/260,907, filed**

## CONTENT AUTHENTICATION AND RECOVERY USING DIGITAL WATERMARKS

### RELATED APPLICATION DATA

[0001] This patent application claims the benefit of U.S. Provisional Application Nos. 60/247,389, filed Nov. 8, 2000, 60/260,907, filed Jan. 10, 2001, and 60/284,594, filed Apr. 17, 2001, which are hereby incorporated by reference.

### TECHNICAL FIELD

[0002] The invention relates to steganography, data hiding, and particularly, digital watermarking of multimedia signals.

### BACKGROUND

[0003] Digital watermarking is a process for modifying physical or electronic media to embed a machine-readable code into the media. The media may be modified such that the embedded code is imperceptible or nearly imperceptible to the user, yet may be detected through an automated detection process. Most commonly, digital watermarking is applied to media signals such as images, audio signals, and video signals. However, it may also be applied to other types of media objects, including documents (e.g., through line, word or character shifting), software, multi-dimensional graphics models, and surface textures of objects.

[0004] Digital watermarking systems typically have two primary components: an encoder that embeds the watermark in a host media signal, and a decoder that detects and reads the embedded watermark from a signal suspected of containing a watermark (a suspect signal). The encoder embeds a watermark by altering the host media signal. The reading component analyzes a suspect signal to detect whether a watermark is present. In applications where the watermark encodes information, the reader extracts this information from the detected watermark.

[0005] Several particular watermarking techniques have been developed. The reader is presumed to be familiar with the literature in this field. Particular techniques for embedding and detecting imperceptible watermarks in media signals are detailed in the assignee's co-pending application Ser. Nos. 09/645,779, filed Aug. 24, 2000, 09/689,293, filed Oct. 11, 2000, 09/503,881 filed Feb. 14, 2000 and U.S. Pat. No. 5,862,260, which are hereby incorporated by reference.

### SUMMARY

[0006] The invention provides methods for embedding hidden data in media signals, and methods for extracting this embedded data to detect alteration of the signal as well as to repair the altered portion of the signal. One aspect of the invention is a method for hiding auxiliary data in a media signal. This method compresses a first media signal, and embeds the first media signal into a second media signal. In one implementation, the second media signal is part of the first signal, and the compressed part of the signal is extracted and used to repair altered portions of the media signal.

[0007] Another aspect of the invention is a method of decoding auxiliary data that has been imperceptibly embedded into a host signal. The method decodes the auxiliary signal, which represents a compressed version of the host

signal. It then decompresses the compressed version, and uses the decompressed version to authenticate the host signal.

[0008] Another aspect of the invention is a method of digitally watermarking media content. The method selects non-overlapping blocks A, B and C of the media content; losslessly compresses blocks B and C of the content to form compressed content; watermark embeds the compressed content into block B of the media content to form a new block B'; creates a hash of block A and B'; watermark embeds the hash into block C to create a new block C'; and combines blocks A, B' and C' to form watermarked media content.

[0009] A compatible watermark decoder compares the hash with a new hash calculated from the watermarked media content to determine whether the watermarked media content is authentic. The decoder reconstructs original un-watermarked media content by using watermark decoding to extract the compressed content and decompressing the compressed content.

[0010] Another aspect of the invention is a method for encoding a reversible watermark in a media signal. The method embeds a watermark message signal into a reference media signal to create a watermarked reference signal, where the watermark message signal includes information about a watermark embedder function used to embed the watermark message signal into the reference signal. The method subtracts the reference signal from the watermarked reference signal to form a difference signal, and adds the difference signal to a host media signal to embed the watermark message signal in the host signal. The adding of the difference signal is reversible by decoding the information about the watermark embedder function from the host media signal, using the information about the watermark embedder function to re-compute the difference signal, and subtracting the difference signal from the host media signal to restore the host signal.

[0011] Another aspect of the invention is a method for authenticating a watermarked media signal that has been watermarked using a watermark embedding function on an un-watermarked version of the media signal. This method decodes a watermark message from the watermarked media signal, wherein the watermark message includes a hash of the un-watermarked media signal and watermark embedding information used by the watermark embedding function to create the watermarked media signal. It transforms the watermark embedding information and hash into a watermark difference signal using the watermark embedding function. It subtracts the watermark difference signal from the watermarked media signal to restore the un-watermarked media signal. The method computes a new hash of the restored, un-watermarked media signal, and compares the new hash with the hash decoded from the watermarked media signal to determine the authenticity of the watermarked media signal.

[0012] Further features of the invention will become apparent from the following detailed description.

## DETAILED DESCRIPTION

## Authentication Watermarking Using Sorting Order Embedding To Embed A Compressed Bit Stream in Another Signal

[0013] This section describes fragile and semi-fragile watermarking methods and related systems and applications. One method embeds a compressed bit stream into a host media signal. In particular, a watermark encoder embeds a compressed version of the host media signal into the host signal. To authenticate a watermarked signal, a compatible decoder extracts the watermark to recover the compressed version and decompresses it.

[0014] The decompressed signal may be compared with the watermarked signal to detect alteration.

[0015] This section also describes a sorting order watermark embedding method and compatible decoder. In this method, an embedder modulates the sorting order of selected sets of samples in a media signal to encode auxiliary information in that signal. The decoder analyzes the sorting order of the sets of samples, and maps the sorting orders to a corresponding message symbols.

[0016] The following sections present a fragile watermarking method for media signal authentication. The method creates a hash of a host media signal, such as a digital image, and embeds the hash back into the host signal. In particular, one implementation employs a compressed bit stream as the hash and embeds the bit stream into the signal by a sorting order embedder.

[0017] While other forms of hashes may be used, compression tools are particularly useful for generating a hash of a signal for use in authenticating that signal. For image signals, the image compression tools (for example, the image compression standard JPEG 2000) available today achieve exceptional compression performance. Even at a very low bit rate, the compression tools provide good image quality, both subjectively and objectively.

[0018] To authenticate a signal using a the compressed signal as a hash, a watermark decoder first extracts the compressed bit stream. Next, the decoder constructs a decompressed signal from the bit stream. By comparison of the signal to be authenticated and the constructed decompressed image, one can tell whether or not the image is authentic or altered.

[0019] If the signal is not authentic, the decoder can also locate the altered region and recover that region from the constructed decompressed signal.

[0020] The following sections illustrate a watermarking implementation for authenticating images. In this implementation, the embedder computes the hash of the host image using the image compression standard JPEG 2000. JPEG 2000 is a wavelet-based image compression system. For more information see, M. J. Gormish, D. Lee, and M. W. Marcellin, "JPEG 2000: Overview, architecture, and applications," in Proceedings of the International Conference on Image Processing, September 2000. D. Taubman, E. Ordentlich, M. Weinberger, G. Seroussi, I. Ueno, and F. Ono, "Embedded block coding in JPEG 2000," in Proceedings of the International Conference on Image Processing, September 2000. W. Zeng, S. Daly, and S. Lei, "Point-wise

extended visual masking for JPEG-2000 image compression," in Proceedings of the International Conference on Image Processing, September 2000.

[0021] JPEG 2000 can compress several different types of still images (bi-level, gray-level, color) with different characteristics (natural images, scientific, medical, military imagery, text, and rendered graphics). Its compression performance is a significant improvement over discrete cosine transform (DCT) based JPEG. G. Wallace, "The JPEG still picture compression standard," Communications of the ACM, vol. 34, no. 4, pp.30-44, 1991.

[0022] In JPEG 2000, a discrete wavelet transform (DWT) first decomposes an image into spatial frequency subbands. See, Stephane Mallat, "A theory of multiresolution signal decomposition: the wavelet representation," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 11, no. 7, pp. 675-693, 1989.

[0023] Each subband is then partitioned into several blocks. Each block is coded independently into embedded block bit stream by the EBCOT algorithm. See, D. Taubman, "High performance scalable image compression with EBCOT," IEEE Transactions on Image Processing, July 2000. The EBCOT algorithm quantizes the wavelet coefficients, groups the quantization bits by context modeling, and uses the MQ binary arithmetic coder to further compress the bit stream. Finally the output of MQ coder from all blocks are collected into packets to form the JPEG 2000 bit stream.

[0024] Besides its exceptional compression performance, JPEG 2000 provides quality scalability, resolution scalability, and spatial scalability. In addition, a lower bit rate image can be constructed by truncating the bit stream of a higher bit rate.

[0025] In fragile watermarking applications, the watermark is designed to detect changes in the signal in which it is embedded. One approach is to create a hash of the host signal and embed the hash back into the host signal. The hash size should not be too large since it will tend to degrade the quality of the signal in which it is embedded. However, to increase the accuracy of authentication, the hash should be as large as possible.

[0026] The implementation makes this tradeoff and takes advantage of the compression performance of JPEG 2000 by using the compressed bit stream from JPEG 2000 as the hash of a digital image.

[0027] The implementation of the embedding process is as follows.

[0028] 1. Compress the digital image to a low bit rate.

[0029] 2. Take the compressed bit stream as a binary message and embed the binary message into the original digital image.

[0030] The implementation of the authentication process is as follows.

[0031] 1. Extract the compressed bit stream from the image.

[0032] 2. Construct a decompressed image from the bit stream.

[0033] 3. Compare the decompressed image and the image to be authenticated.

[0034] 4. If the image is found altered, locate and recover the altered region by the constructed decompressed image.

[0035] A detailed example of the embedding and authentication processes is explained below.

[0036] As an example, we apply the watermarking method presented above to an image. In this example, the image is compressed at a compression ratio 32:1 and the compressed bit stream is embedded in the spatial domain. As such, on average, one message bit of information is embedded into every four pixels.

[0037] The embedder uses a sorting order method to embed one message bit into a 2 by 2 block of pixels. The sorting order embedder sorts all pixels values in the block. In a 2 by 2 block, there are 81 possible sorting orders. The embedder partitions all possible sorting orders into two groups, S1 and S0. If the message bit to be embedded is 0, and the current sorting order is in S0, no action is needed. If the sorting order is in S1, the embedder modifies these four pixel values so that the new sorting order will be in S0. The message bit "1" is embedded similarly.

[0038] After each of the compressed bit stream is embedded, the result is a watermarked image in which the watermark is imperceptible.

[0039] To authenticate the watermarked image, the compressed bit stream is extracted from the sorting order decoder and a decompressed image is constructed. If the watermarked image has been modified after watermarking, the modification can be detected by comparing the image to be authenticated and the decompressed image. In addition, the altered region can be located and recovered from the decompressed image.

[0040] Based on the preceding description, the following points can be made:

[0041] In the JPEG 2000 bit stream, some header information (such as the size of the image) can be discarded since it can be set to the same size as the host image being authenticated. This will reduce the hash size.

[0042] As an alternative to embedding in blocks of samples in the spatial domain, the compressed bit stream can be embedded in a transform domain as well, such as the wavelet domain, Fourier domain, DCT, etc.

[0043] The sorting order embedder is not restricted to embedding auxiliary data symbols in the spatial domain. It can also be employed in a transform domain.

[0044] The compressed bit stream may be modulated by a pseudo random binary sequence before it is embedded.

[0045] The watermarking method can be modified to be robust against JPEG 2000 compression (up to a chosen compression ratio). Since the JPEG 2000 is an embedded coding, a lower bit rate image can be constructed by truncating the bit stream of a higher bit rate. Thus, the embedder can embed the hash into

the selected quantization bits in the wavelet domain, and an inverse DWT will give the watermarked image.

[0046] The embedding and decoding of the watermarking method can be done blockwise. In such an approach, the blockwise embedder divides the host signal into N by M blocks, compresses each block and embeds the bit stream into another block.

[0047] This method can also be used to hide one media signal into another (such as one image into another).

[0048] The compression process can be used to encode other types of watermark messages. For example, if the watermark is a binary message, one could compress the binary message by entropy coding to reduce its size (thus increases the hiding capacity). Error correction code (ECC) such as convolution codes or BCH codes may be used to encode the watermark message. If an error correction coding is applied to the binary message before embedding, then the compression will be employed before ECC.

#### Authentication Watermarking Enabling Recovery of the Original Un-watermarked Content

[0049] The disclosure describes a method of watermarking content, particularly images, in a manner that enables authenticity of the content to be verified (e.g., determine whether the content has been altered) and also enables the original un-watermarked content to be re-constructed.

[0050] The following description details a watermark embedding method, a corresponding authentication and recovery method, and an example implementation. The watermark embedding method is as follows:

[0051] [1] Divide the digital image I into three non-overlapping regions A, B, and C. For example, A could be a spatial block, a frequency band, the six most significant bits, etc.

[0052] [2] Losslessly compress B and C, and use a watermark encoding process to store the compressed bit stream in B, which results in a new B'. Examples of lossless compression techniques include, for example, entropy coding methods like Huffman coding, arithmetic coding, etc.

[0053] [3] Create a hash of A and B', and use a watermark encoding process to store the hash in C, which results a new C'. Examples of hashes include secure hashes like SHA-1.

[0054] [4] Combining A, B', and C' gives the watermarked image.

[0055] The authentication and recovery method is as follows:

[0056] [1] For a digital image i to be authenticated, divide it into three non-overlapping regions a, b, and c, as in the first step of embedding.

[0057] [2] Create the hash of a and b, and compare it with the hash decoded from c using a watermark decoding process compatible with the encoding process. If they are exactly the same, i is classified as authenticated.

[0058] [3] If  $i$  is authenticated, decompress the watermark embedded into  $b$  by first decoding the watermark in  $b$  and then decompressing the watermark to derive  $b'$  and  $c'$ .

[0059] [4] Combine  $a$ ,  $b'$ , and  $c'$ . We claim it is the original, unwatermarked image.

[0060] One implementation of the above (embedding) method is as follows:

[0061] [1] We define a haar transform of two integers  $m$  and  $n$

[0062]  $a=\text{round}((m+n)/2)$ ,  $b=m-n$ .

[0063] For example, if  $m=150$ ,  $n=81$ , we have

[0064]  $a=\text{round}((150+81)/2)=\text{round}(231/2)=116$

[0065]  $b=150-81=69$

[0066] As it can be easily seen, we can retrieve  $m$  and  $n$  from  $a$  and  $b$ .

[0067] [2] For a digital image, we apply the haar transform row by row, then column by column, as like a one level wavelet transform. We can also define the low frequency  $L$  (from the  $a$  part) and high frequency  $H$  (from the  $b$  part), as in a wavelet domain.

[0068] [3] We take the  $LL$  (the average part from  $a$ ) as region A.

[0069] [4] We take the last 160 bits from  $HH$  as region C.

[0070] [5] We take the  $LH$ ,  $HL$ , and  $HH$  except the last 160 bits as region B.

[0071] [6] Arithmetic coding on all bits from  $LH$ ,  $HL$ , and  $HH$ , and store the compressed bit stream in B.

[0072] [7] Run the secure hash algorithm on A and B, and generate a 160-bit hash.

[0073] [8] Store the 160-bit hash in C.

[0074] Appendix A includes source code in MatLab format for the watermark embedding and decoding implementation described above.

[0075] Before applying lossless compression in [6], the implementation prepares the integer coefficients,  $x$ , for lossless compression as follows:

[0076]  $\text{Sign}(x)=1$ , when  $x>0$ .

[0077]  $\text{Sign}(x)=0$ , when  $x<0$ .

[0078] The absolute value of  $x$  ( $\text{abs}(x)$ ) is set to  $\text{abs}(x)+1$ . The resulting number is represented as a binary sequence,  $b_1b_2\dots b_n$ , where  $b_1=0,1$ . Since  $b_1$  is always 1, the implementation does not code  $b_1$ .

[0079] In other words,

[0080]  $b_1b_2\dots b_n$  is represented as  $b_2\dots b_n$ .

[0081] The new binary string,  $b_1b_2\dots b_n$ , is used as input to the lossless coding process, which in this implementation is an arithmetic coder described in "Arithmetic coding for data compression", I. H. Witten, R. Neal, and J. G. Cleary, Communications of the ACM, 30:520-540, June 1987, which is hereby incorporated by reference. An example

source code implementation of the arithmetic coder is provided in this paper. In decoding, the inverse of the lossless coding method is used to reconstruct  $b_2\dots b_n$ , which is then used to reconstruct  $x$  by performing the inverse of the method described above.

#### Content Authentication and Recovery from Tampering Using Embedded Compressed Bit Streams

[0082] This section describes a method for authenticating a media signal and for restoring regions of the signal when tampering has been detected.

[0083] To illustrate the method, we use the example of a digital image, though it also applies to other media signal types like audio and video signals. The method determines whether local regions of an image have been altered, and if so, recovers the tampered regions.

[0084] There are two components to the method: the embedder and the detector. The embedder calculates a hash of an input digital image, and embeds the hash into the digital image. The detector re-computes the hash of a suspect image, extracts the hash from the image, and finally, compares the re-computed and extracted hashes. If they differ, it proceeds to recover tampered regions. A more detailed description of an implementation follows below.

[0085] The embedder partitions a digital image into two non-overlapping regions. The first region is relatively large compared to the second, and contains most of the perceptually important visual content. The second region is smaller and contains almost no perceptually important visual content.

[0086] The embedder embeds a hash of the first region into the second region. Preferably the hash should have the following attributes:

[0087] Unique—identical images produce identical hashes

[0088] Capacity—the hash should be small so that it will not degrade the quality of the image when embedded into the second region

[0089] Confidence—if two images have the same hash, they should be perceptually similar (no visual differences)

[0090] Invariance—the hash of the image before and after embedding should be the same. In one implementation, the embedder divides the image into blocks, then partitions each block into regions (namely, region I and region II). One example partitioning is to separate the most significant bits from the least significant bits (e.g., for an 8 bit per pixel image, region I comprises the 7 most significant bits and region II comprises the least significant bit. Next, it compresses region I of each block to compute a hash for that block. In particular, it performs a JPEG 2000 compression of region I. It then embeds the compressed bit stream of region I of a block (a source block) into region II of one or more different blocks (the target block(s)). This process is repeated for each block such that each block carries the compressed bit stream of one or more other blocks.

[0091] One example implementation uses a JPEG 2000 compression ratio of 32:1 to create the hash, but the actual compression ratio chosen can vary depending on data hiding capacity, image quality, and confidence level. For added security, the hash (namely, the compressed bit stream) is modulated by a pseudo random binary sequence.

[0092] The embedding process proceeds as follows:

[0093] 1. partition region II of a block into N sub-regions;

[0094] 2. for each source block, compute a fragile hash of the compressed bit stream of the source block;

[0095] 3. for each source block, compute N permutations, where the permutations map N-1 instances of the compressed bit stream of the source block and the fragile hash of the source block to N corresponding sub-regions in target blocks;

[0096] 4. Embed the compressed bit streams and fragile hash into region II of selected target blocks by replacing bits in corresponding sub-regions of region II with N-1 permutations and one fragile hash.

[0097] The fragile hash is referred to as "fragile" because even slight modifications of the data input to the hash create different hash outputs. Examples include MD4, MD5, SHS, a check sum, etc. In one implementation, the embedder computes the fragile hash of the modulated, compressed bit stream for the source block.

[0098] The number of sub-regions of region II may vary. In one implementation where the compression ratio is 32:1, we use 4 sub-regions. As such, there are three redundant instances of the compressed bit stream for each block, and one fragile hash. Generally speaking, the embedder embeds repeated instances of a first hash (the compressed bit stream), and one or more instances of a fragile hash of the first hash. We refer to the first hash as the "redundant hash."

[0099] To authenticate a suspect image, a detector reverses the embedding process. It proceeds as follows:

[0100] 1. Divides the suspect image into blocks as in the embedder;

[0101] 2. Partition each block into region I and region II as in the embedder;

[0102] 3. In each block, compress region I (JPEG 2000 compression at the same compression ratio as the embedder);

[0103] 4. Modulate the compressed bit stream of step 3 with the same pseudo random number sequence used in the embedder;

[0104] 5. Use the permutation to find the target blocks carrying the N-1 versions of the compressed bit stream for each source block, and the fragile hash of the source block; and

[0105] 6. Evaluate the authenticity by comparing the N-1 versions of the compressed bit stream and fragile hash.

[0106] There are a number of possible schemes for verifying the authenticity using the redundant hashes (e.g., the compressed bit streams) and the fragile hash (the hash of the

redundantly embedded hash). In one implementation, the detector compares each of the extracted redundant hashes for a block with the re-computed hash for that block and signals that tampering has occurred when at least one of the extracted hashes does not match the re-computed hash.

[0107] In this implementation, there are three possibilities when the detector finds tampering:

[0108] 1. one of the N-1 target blocks or the source block (excluding its LSBs) has been tampered;

[0109] 2. The LSBs of one of the N-1 target blocks has been tampered; or

[0110] 3. Both 1 and 2 have occurred.

[0111] The fragile hash of the source block serves as a guide for recovering the original image data of the tampered target block. When there is a mis-match between the re-computed version of the redundant hash and one or more of the embedded redundant hashes for a source block, the detector extracts the embedded fragile hash and compares it with the fragile hash of each of the redundant hashes for the source block as well as the fragile hash of the re-computed hash. If the embedded fragile hash matches the fragile hash of the re-computed hash of the source block, then the MSBs of the source block have not been altered. However, tampering has occurred in the LSBs of the target blocks for which the fragile hash does not match.

[0112] If the embedded fragile hash does not match the fragile hash of the source block, but does match the fragile hash of one or more of the embedded hashes for that block, then the MSBs of the source block have been altered. Any one of the compressed bit streams with matching fragile hashes can then be used to reconstruct the MSBs of the source block. In particular, the compressed bit stream extracted from one of the target blocks is decompressed and used to replace the MSBs of the source block.

[0113] If the embedded fragile hash does not match the fragile hash of the re-computed hash or the fragile hash of any of the embedded redundant hashes, then the detector forms a new bit stream for the source block by combining the compressed bit stream for that block and the embedded bit streams for that block. One way to combine these data sets is to pick the bit value for each bit that occurs most frequently. If the compressed bit stream for the source block does not match the stream formed by combining these data sets, then the stream formed from combining the data sets can be decompressed to reconstruct an approximation of the original source block.

[0114] There are number of possible variations to this process. For example, the redundant hashes for a particular block can be used as an error correction code, where the redundant hashes are extracted from the target blocks and combined to form a single composite hash. This composite hash can be used to reconstruct the source block when the detector determines that it has been altered using the analyses described above. Either the composite hash or the individual instances of the redundant hash may be compared with the recomputed hash to detect alteration. Further, the extracted fragile hash can then be compared with the fragile hash of the source block, and the fragile hash of the composite hash or the individual instances of the redundant hash to detect alteration.

[0115] While the above technique is illustrated for images, it may also be applied to video and audio signals. For example, the same technique may be applied to video frames. Also, the technique may be applied to a digital audio signal by segmenting the audio signal into blocks, partitioning those blocks into regions, and compressing the first region of each block and embedding it into the second region of one or more other blocks.

[0116] The above technique may also be applied to compressed bit streams representing images, video or audio. For example, the compressed bit stream is segmented into blocks, and the blocks are partitioned into regions, one region containing perceptually relevant information, and another region containing less perceptually relevant information. A hash or compressed version of the first region of each block is then embedded into one or more of the second regions of other blocks.

#### Reversible Watermarking and Authentication of Media Signals

[0117] The following description details a digital watermarking process where the watermark embedding process is reversible from the standpoint that it enables recovery of the original unmodified host signal. The particular method described below inserts information about the embedding function into a watermark message payload. The payload includes two components: the watermark message (which includes a hash of the host image for authentication applications) and the embedding function information. In the watermark detection stage, when the payload is successfully decoded, it unveils the embedding function information. Based on such information, the watermarked image (or other media content like audio, video) undergoes the reverse of the embedding procedure to remove the watermark and derive the original, un-watermarked image.

[0118] The following is an example of an implementation for a grayscale image using a watermark embedder plugin of the Adobe Photoshop program from Digimarc Corporation:

#### [0119] Embedding:

[0120] [1] For an image X, create a hash H of X. The hashing function could be the secure hashing algorithm, (SHA-1), 2-D hashing, compressed bit stream, etc.

[0121] [2] Create a mid-gray image G with all pixel values being 128. This image G serves as a reference signal.

[0122] [3] Watermark G using the Adobe Photoshop watermark embedder plugin, where the payload consists of H and the watermark durability. If the watermark durability is fixed at some constant known to the detector, e.g., 1 (or another constant), then the watermark durability does not need to be put into payload. Assume the watermarked image is G'.

#### [0123] [4] Define

$$X' = X + G' - G$$

[0124] The subtraction of the reference signal G from the watermarked reference signal G' gives a watermark difference signal, D=G'-G.

[0125] As an alternative to steps [1-4], the implementer may create a difference signal by the following steps:

[0126] [a] form the watermark message from the hash, watermark embedder information (a gain value), and error detection bits (e.g., CRC bits),

[0127] [b] perform error correction encoding of the message (such as convolution encoding, BCH encoding, etc.),

[0128] [c] multiplying or XORing the resulting binary number with a binary antipodal pseudorandom sequence generated from a key seed number to form a spread spectrum signal, and

[0129] [d] mapping the spread spectrum signal into a two dimensional image and multiplying the image pixel values by the gain value to form a watermark difference signal D.

[0130] [5] X' will be the watermarked image of X.

#### [0131] Detection:

[0132] [1] Use the Digimarc watermark reader plugin in Adobe Photoshop to detect a possibly watermarked image X'.

[0133] Alternatively, the implementer may create a watermark reader compatible with the alternative embedder described above:

[0134] [a] cross correlate the watermarked image with the pseudorandom sequence mapped into an image to decode estimates of error correction encoded bits.

[0135] [b] perform error correction decoding of the message bit estimates (such as convolution encoding, BCH encoding, etc.).

[0136] [c] use decoded error detection bits to check for errors to determine if valid message is present. If so, the detection process has succeeded.

[0137] [2] If [1] of the detection process has succeeded, watermark the mid-gray image G with the decoded payload. Assume the watermarked image is G''. Alternatively, create a watermark difference signal, D', from the decoded payload using embedder steps [4][a-d].

#### [0138] [3] Define

$$X'' = X + G'' - G' \quad (X'' = X - D')$$

[0139] [4] Compare the hash from the decoded payload with the hash of X''.

[0140] [5] If the hash match exactly, then X'' (and equivalently X') is authenticated and X'' is also the original, un-watermarked image.

[0141] The discrete values of a digital media signal are usually bounded in a fixed range. For example, in an eight bit grayscale image, the pixel value is an integer between 0 and 255. Thus, when the embedder adds a watermark difference signal D to an image X, it takes into account the overflow and underflow problems. The overflow refers to the case when the pixel value is above the upper limit (which will be larger than 255 in an eight bit grayscale image); the underflow refers to the case when the pixel value is below the lower limit (which will be lower than 0 in an eight bit grayscale image).

[0142] In the embedding stage, when an overflow (or underflow) happens at some pixel, the embedder records the pixel location and the overflow (or underflow) value, which is the difference between the upper (or lower) limit and the pixel value. For example, after adding the watermark difference signal D, if the pixel value x' is 256, then the embedder records the location of x' and the overflow value 1, and changes the value of x' to 255; Similarly if the pixel value x" is -1, then the embedder records the location of x" and the underflow value -1, and changes the value of x" to 0.

[0143] The watermark embedding is processed block by block, where the image is subdivided into contiguous rectangular blocks of pixels. The overflow and underflow locations and values of one block are stored in the payload of another block. Such overflow and underflow information can also be employed for authentication, since the pixel values at overflow locations must be 255 in the case of eight bit grayscale, and the pixel values at underflow locations must be 0 in the case of eight bit grayscale.

[0144] In one implementation, the overflow and underflow pixel locations and values for a first image block are included as part of the watermark embedder information in the watermark payload embedded in a second image block, such as a neighboring block, or a block that has a relationship with the first block specified by a key. This key may be an index to the first block embedded in the second block, or an offset specifying the position of the block from which the data originated. At decoding, the decoder decodes the payload information, including the overflow/underflow information, and uses this information to restore the original image after decoding step [3] and before authentication step [4].

[0145] After correcting for overflow/underflow errors for a particular block, the decoder then recomputes the hash of the restored image block and compares it with the hash decoded from the watermark payload. If the hash does not match, the image block is not authentic. Otherwise, it is deemed authentic. This process is repeated for other watermarked blocks in the image to check their authenticity.

[0146] A watermarked block of samples may carry the overflow/underflow data for more than one other block, as long as the data in that watermark payload specifies from which other block that data originated (e.g., using a block index or offset). Also, if the watermark embedder strength is uniform across an image or part of an image, it need not be carried in every block. Instead, one block can carry the strength parameter that applies to many blocks, along with a parameter indicating the blocks to which that parameter should be applied. In this manner, watermark control parameters for two or more blocks may be stored in the watermark payload of a single block. This block based embedding of control parameters enables the watermark control parameters to vary from block to block across the host signal.

[0147] This technique may be extended to other media types, including audio and video, as well as other watermark embedder/detector methods.

#### Concluding Remarks

[0148] Having described and illustrated the principles of the technology with reference to specific implementations, it

will be recognized that the technology can be implemented in many other, different, forms. For example, the method may be applied to other data types, like audio, video and geometric models for computer generated graphics, etc. To provide a comprehensive disclosure without unduly lengthening the specification, applicants incorporate by reference the patents and patent applications referenced above.

[0149] While the invention is illustrated with reference to images, it also applies to other media types including audio. In the case of audio, the signal hash may be computed from and embedded into temporal blocks of an audio signal. The watermark embedding may modulate features in the time, frequency, or some other transform domain of the host audio signal block.

[0150] In addition to a hash, the watermark may be used to convey other information, such as an identifier of the content, an index to related metadata, rendering control instructions, etc. For example, the watermark can carry a network address or index to a network address to link the watermarked signal to a network resource such as a related web site. Some blocks may be used to carry a hash, while others may be used to carry other payload information, such as metadata, or a pointer to metadata stored in an external database.

[0151] The methods, processes, and systems described above may be implemented in hardware, software or a combination of hardware and software. For example, the auxiliary data encoding processes may be implemented in a programmable computer or a special purpose digital circuit. Similarly, auxiliary data decoding may be implemented in software, firmware, hardware, or combinations of software, firmware and hardware. The methods and processes described above may be implemented in programs executed from a system's memory (a computer readable medium, such as an electronic, optical or magnetic storage device).

[0152] The particular combinations of elements and features in the above-detailed embodiments are exemplary only; the interchanging and substitution of these teachings with other teachings in this and the incorporated-by-reference patents/applications are also contemplated.

---

embed\_wm.txt

```

function y = embed_wm(x);
% y = embed_wm(x)
%
%File Name: embed_wm.m
%Last Modification Date: Wed Nov 1 2000
%File Creation Date: Wed Nov 1 2000
%Author: Jun Tian <jian@digimarc.com>
%
%Copyright: All software, documentation, and related files in this distribution
%           are Copyright (c) 2000 Digimarc Corporation
%
%Change History:
%
%Bug Report: mail to jian@digimarc.com
%
x = haar(x);
[m,n] = size(x);
bits = 0;
for i = 1:m
    for j = (n/2+1):n
        if (x(i,j) >= 0)

```

-continued

```

        bits = [bits +];
    else
        bits = [bits -]
    end
    a = abs(x(i,j)) + 1;
    a = a - 2 * (floor(log(a)/log(2)));
    while (a > 0)
        bits = [bits (a - 2*floor(a/2))];
        a = a - 2 * floor(a/2);
    end
end
for i = (m/2+1) :m
    for j = 1:(n/2)
        if (x(i,j) >= 0)
            bits = [bits +];
        else
            bits = [bits -];
        end
    end
    a = abs(x(i,j)) + 1;
    a = a - 2 * (floor(log(a)/log(2)));
    while (a > 0)
        bits = [bits (a - 2*floor(a/2))];
        a = a - 2 * floor(a/2);
    end
end
bits = [bits SHA-1(x(1:(m/2),1:(n/2)))];
bits = bits(2:length(bits));
b = arithmetic_coding(bits);
k = 1;
for i = 1:m
    for j = (n/2+1) :n
        q = max_capacity(x(i,j));
        a = 1
        for l = 1:q
            a = a*2 + b(k);
            k = k + 1;
        end
        if (sign(x(i,j)) >= 0)
            x(i,j) = a;
        else
            x(i,j) = -a;
        end
    end
end
for i = (m/2+1) :m
    for j = 1:(n/2)
        q = max_capacity(x(i,j));
        a = 1
        for l = 1:q
            a = a*2 + b(k);
            k = k + 1;
        end
        if (sign(x(i,j)) >= 0)
            x(i,j) = a;
        else
            x(i,j) = -a;
        end
    end
end
y = ihaar (x);
function y = haar(x);
% y = haar(x)
%
%File Name: haar.m
%Last Modification Date: Wed Nov 1 2000
%File Creation Date: Wed Nov 1 2000
%Author: Jun Tian <jtian@digimarc.com>
%
%Copyright: All software, documentation, and related files in this distribution
%           are Copyright (c) 2000 Digimarc Corporation

```

-continued

```

%
%Change History:
%
%Bug Report: mail to jtian@digimarc.com
%
[m,n] = size(x);
z(1:(m/2),:) = ceil((x(1:2:m,:) + x(2:2:m,:))/2);
z((m/2+1):m,:) = x(1:2:m,:) - x(2:2:m,:);
y(:,1:(n/2)) = ceil((z(:,1:2:n) + z(:,2:2:n))/2);
y(:,(n/2+1):n) = z(:,1:2:n) - z(:,2:2:n);
ihaar.txt
function y = ihaar(x);
% y = ihaar(x)
%
%File Name: ihaar.m
%Last Modification Date: Wed Nov 1 2000
%File Creation Date: Wed Nov 1 2000
%Author: Jun Tian <jtian@digimarc.com>
%
%Copyright: All software, documentation, and related files in this distribution
%           are Copyright (c) 2000 Digimarc Corporation
%
%Change History:
%
%Bug Report: mail to jtian@digimarc.com
%
[m,n] = size(x);
z(:,1:2:n) = floor((2*x(:,1:(n/2)) + x(:,(n/2+1):n))/2);
z(:,2:2:n) = floor((2*x(:,1:(n/2)) - x(:,(n/2+1):n))/2);
y(1:2:m,:) = floor((2*z(1:(m/2),:) + z((m/2+1):m,:))/2);
y(2:2:m,:) = floor((2*z(1:(m/2),:) - z((m/2+1):m,:))/2);

```

I claim:

1. A method for hiding auxiliary data in a media signal, the method comprising:

compressing a first media signal; and

embedding the first media signal into a second media signal.

2. The method of claim 1 wherein the first media signal is at least a part of the second media signal.

3. The method of claim 1 wherein the embedding includes associating a symbol with a sorting order, determining a sorting order of a block of samples, and modulating the sorting order of the block of samples as necessary to make the sorting order match the sorting order associated with a symbol to be embedded.

4. A computer readable medium on which is stored software for performing the method of claim 1.

5. A method of decoding auxiliary data that has been imperceptibly embedded into a host signal:

decoding the auxiliary signal, which represents a compressed version of the host signal;

decompressing the compressed version;

using the decompressed version to authenticate the host signal.

6. The method of claim 5 wherein the decoding includes using a sorting order decoder to analyze sorting order of selected blocks of samples, and to look up a symbol corresponding to the sorting order.

7. A computer readable medium on which is stored software for performing the method of claim 5.

8. The method of claim 5 wherein the decoding is performed on blocks of the host signal.

9. The method of claim 8 wherein auxiliary data decoded from one block of the host signal is used to authenticate another block of the host signal.

10. A method of watermarking media content comprising:

- selecting non-overlapping blocks A, B and C of the media content;
- losslessly compressing blocks B and C of the content to form compressed content;
- watermark embedding the compressed content into block B of the media content to form a new block B';
- creating a hash of block A and B';
- watermark embedding the hash into block C to create a new block C'; and
- combining blocks A, B' and C' to form watermarked media content.

11. The method of claim 10 wherein the media content is an image.

12. The method of claim 10 wherein the media content is an audio signal.

13. A computer readable medium having software for executing the method of claim 10.

14. A watermark decoder for decoding the hash from the watermarked media content created by the method of claim 10, the decoder operable to compare the hash with a new hash calculated from the watermarked media content to determine whether the watermarked media content is authentic.

15. The watermark decoder of claim 14 further including a decoder for reconstructing original un-watermarked media content by using watermark decoding to extract the compressed content and decompressing the compressed content.

16. A computer readable medium on which is stored software for implementing the watermark decoder of claim 14.

17. A method for encoding a reversible watermark in a media signal comprising:

- embedding a watermark message signal into a reference media signal to create a watermarked reference signal, where the watermark message signal includes information about a watermark embedder function used to embed the watermark message signal into the reference signal;
- subtracting the reference signal from the watermarked reference signal to form a difference signal; and
- adding the difference signal to a host media signal to embed the watermark message signal in the host signal;

wherein adding the difference signal is reversible by decoding the information about the watermark embedder function from the host media signal, using the information about the watermark embedder function to re-compute the difference signal, and subtracting the difference signal from the host media signal to restore the host signal.

18. The method of claim 17 further including:

- computing a hash of the host signal; and including the hash in the watermark message signal such that the message signal includes the hash and the information about the watermark embedder function.

19. The method of claim 17 wherein the information about the watermark embedder function is a watermark durability parameter.

20. The method of claim 17 wherein the host signal is a still image.

21. The method of claim 20 wherein the reference signal is an image of substantially uniform pixel values.

22. The method of claim 21 wherein the reference signal comprises uniform pixel values at a mid-level between a range of possible pixel values between a lowest possible value and a highest possible value.

23. A computer readable medium on which is stored software for performing the method of claim 17.

24. A method for authenticating a watermarked media signal that has been watermarked using a watermark embedding function on an un-watermarked version of the media signal, comprising:

- decoding a watermark message from the watermarked media signal, wherein the watermark message includes a hash of the un-watermarked media signal and watermark embedding information used by the watermark embedding function to create the watermarked media signal;
- transforming the watermark embedding information and hash into a watermark difference signal using the watermark embedding function;
- subtracting the watermark difference signal from the watermarked media signal to restore the un-watermarked media signal;
- computing a new hash of the restored, un-watermarked media signal; and
- comparing the new hash with the hash decoded from the watermarked media signal to determine the authenticity of the watermarked media signal.

25. The method of claim 24 wherein transforming includes watermark embedding a reference signal with the hash and watermark embedding information to compute a watermarked reference signal, computing a difference signal between the watermarked reference signal and the reference signal to compute the watermark difference signal.

26. The method of claim 25 wherein the reference signal comprises an image having substantially uniform pixel values.

27. A computer readable medium on which is stored software for performing the method of claim 24.

28. A method for encoding a reversible watermark in a media signal comprising:

- creating a watermark difference signal carrying information about a watermark embedder function used to embed the watermark message signal into the difference signal; and
- adding the difference signal to a host media signal to embed the watermark message signal in the host signal;

wherein adding the difference signal is reversible by decoding the information about the watermark embedder function from the host media signal, using the information about the watermark embedder function to

re-compute the difference signal, and subtracting the difference signal from the host media signal to restore the host signal.

**29.** A computer readable medium on which is stored software for performing the method of claim 28.

**30.** A method of hiding auxiliary data in a media signal, the method comprising:

dividing the media signal into blocks;

partitioning the media signal into two regions;

for a plurality of the blocks, compressing the media signal from a first region of a block and embedding redundant instances of the compressed media signal of the block into a second region of two or more blocks.

**31.** The method of claim 30 including:

for a plurality of blocks, computing a fragile hash of the first region of the media signal for the block and embedding the fragile hash into a second region of one or more blocks.

**32.** The method of claim 31 wherein the fragile hash comprises a fragile hash of the compressed media signal of the block.

**33.** The method of claim 31 wherein the second region of blocks in the media signal are partitioned into sub-regions and the sub-regions for the blocks are embedded with a fragile hash from another block and instances of the compressed media signal from the first region of other blocks.

**34.** A computer readable media on which is stored software for performing the method of claim 30.

**35.** The method of claim 30 wherein the second region comprises one or more least significant bits of data samples of the media signal, and embedding of redundant instances comprises replacing least significant bits of the data samples.

**36.** The method of claim 30 wherein the redundant instances are scrambled before the embedding using a key.

**37.** The method of claim 30 wherein the redundant instances are mapped to different blocks for embedding according to a permutation.

**38.** A method of authenticating a media signal using hidden embedded data in the media signal, the method comprising:

dividing the media signal into blocks;

partitioning the blocks into regions;

for a plurality of blocks, extracting hidden compressed bit streams of a first region other blocks from a second region in the blocks;

for a plurality of blocks, evaluating whether a block is altered by comparing the extracted compressed bit streams for a block with the media signal in the first region of the block; and

when an altered block is detected by the comparison, using a fragile hash to identify location of altered data, and using an extracted compressed bit stream to replace the altered data.

\* \* \* \* \*